Internet Computer as a Data Availability Layer

Dor Cohen
TU Berlin

Yvonne-Anne Pignolet
DFINITY Foundation

Ognjen Maric
DFINITY Foundation

Stefan Schmid
TU Berlin

***Abstract*—Layer-2 networks are widely used to optimize throughput, latency, and costs by aggregating transactions and posting them, batched, to more established blockchains such as Bitcoin and Ethereum. Further reduction in blockspace usage is achievable by off-loading the actual transaction data to an external storage medium, while posting only a reference to the chain. Ensuring that off-loaded data is available is not a trivial task and it is referred to as the Data Availability (DA) problem. Currently, Arbitrum Nitro supports the off-chain storage through a set of trusted nodes called Data Availability Committee. In this work, we leverage smart contracts on the Internet Computer network as a storage layer for transaction batches produced by Arbitrum Nitro. Our implementation introduces minimal modifications to Arbitrum Nitro and is realized as a native extension. For the first time, we demonstrate the feasibility of using the Internet Computer as a DA layer, enhancing the usability, latency, and cost efficiency of provisioning and maintaining a decentralized storage network. Consequently, the Internet Computer (IC) can position itself as a canonical solution to the DA problem.**

***Index Terms*—Arbitrum, Blockchain, Data Availability, Internet Computer, Layer 2**

I. Introduction

Blockchains such as Bitcoin and Ethereum are renowned for their high level of security. However, their adoption often comes with significantly higher usage costs compared to contemporary blockchains. A recent improvement to the Ethereum blockchain, EIP-4844, reduces storage costs by introducing a specialized data storage type called a Blob. This change aligns with the Ethereum community's "Layer-2 centric" vision, making Ethereum more attractive as a Layer-1 [1].

Layer-2 (L2) networks process transactions but depend on a larger, more established blockchain—typically referred to as a Layer-1 blockchain (L1)—for transaction finalization. L2 networks act as transaction processors and batchers. They aggregate, validate, order, and batch transactions before passing them to the L1. In the world of L2 blockchains, there are two main categories. The first one is based on zero-knowledge proofs, ensuring that transaction execution is easily verifiable. The second category, optimistic processing, involves posting batched and processed transactions to the L1 chain without an accompanying proof of correctness. Instead, verifier nodes are responsible for checking the validity of transactions within a block. If an invalid transaction is detected, they can raise an alert and penalize malicious actors.

Optimistic L2s, such as Arbitrum, produce multiple L2 blocks every second, each containing dozens of transactions. To maintain the same level of security as the L1 chain, these blocks must be posted to the L1, regardless whether optimistic or zk-based chains. Without a cost-effective mechanism for block storage, high transaction costs become inevitable, and can deter users or place significant storage pressure on the L1. For example, blocks posted by Arbitrum to Ethereum before EIP-4844 incurred up to 100 times the costs [2]. If Ethereum is to serve as a central, global hub that attracts increasing usage over time, further optimization is essential. Fortunately, transaction data storage can be off-loaded to external, distributed storage networks, often referred to as Data Availability layers.

Data Availability layers are typically divided into two types. The first type, *Data Availability Committee (DAC)*, consists of a group of publicly accessible nodes that store data. Clients requesting data typically query each committee member and accept the data once a sufficient threshold of valid responses is met. In contrast, *Data Availability Sampling* relies on periodically sampling full nodes on a specialized blockchain tailored for transaction storage. Light clients can efficiently verify the availability of data on the DA layer using erasure codes. Reed-Solomon encoded data enables high confidence in data availability (greater than 99.5%) with only $\sqrt{n}$ samples taken, making this approach highly efficient [3]. Arbitrum Nitro recently introduced support for the AnyTrust mode, in which the transaction batches are submitted to a DAC. Each DAC member's responses are collected and validated by the requesting client, while the stored data is considered public and must be accessible at any time. Ethereum persists only a reference to that data containing the location and the public keys used to validate it. This approach offers significant advantages—storage is cheap and performant, and interacting with a DAC is as straightforward as with any other web service.

However, challenges can arise when relying on a DAC. It is expected to have a suitable infrastructure for the DAC upon a creation of a new Nitro chain. A chain owner has to either provision nodes to be acting as committee members, or alternatively resort to use known node operators [6]. In either case there is a need to provision and maintain the DA layer itself. Furthermore, good behavior is not incentivized and nodes can misbehave by withholding blocks, while the course of action is left to the discretion of the chain owner. As mentioned, prior to AnyTrust, data was posted to Ethereum, serving as a reliable and available source. When transaction data is detached and stored off-chain, DAC member misbehavior can cause issues to the proper operation of the L2. In the case of optimistic processing, validators would be unable to retrieve and validate transactions. Thus, in order to preserve the correctness of data on the chain, it would halt, and the affected transactions are

to be reversed in order to bring the chain to a correct state.

These issues are evident, leading us to question whether a better approach is possible. Can we replicate the functionality of a DAC in a cost-effective, ergonomic and decentralized manner?

II. IC as a DA Layer

Our proof of concept [7] presents a simple approach - let us delegate storage to the Internet Computer [4], a blockchain capable of storing large amounts of data. We utilize canisters on the Internet Computer to store batches from Arbitrum Nitro, effectively simulating a decentralized DAC. A canister is similar to a smart contract, running on all replicas within a subnet and can hold up to 500GB of data. A subnet is a partition of replicas reaching a consensus regarding the state of those canisters. Subnets can vary in size, depending on their purpose. Application subnets host standard user canisters such as our DA application and typically contain about one dozen nodes, offering a fair trade-off between performance and distribution, as consensus is faster and easier to achieve with this cluster size. The control over a replica's membership in a subnet is decided by an overarching governance system. The governance logic itself is also implemented as canisters; however, unlike standard canisters, these are considered special and run on a much larger subnet called the Network Nervous System (NNS). The weight of each vote for governance proposals is proportional to the amount of governance token (ICP) staked by each actor. Replica operators are compensated by predictable payouts of ICP tokens, while voters are compensated for governance participation [8]. This tandem between execution replicas and global governance allows for *deterministic decentralization* [9].

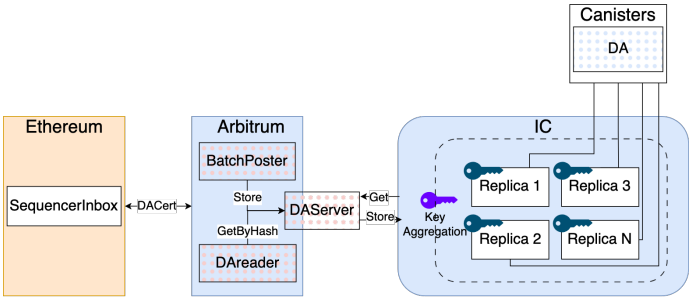

Fig. 1. **Internet Computer as a DA layer for Arbitrum Nitro**
The dotted components are the core of our implementation. Red dotted components are either modifications or native extensions of Arbitrum Nitro, while the blue dotted one (DA canister) is the the storage logic on IC. The dashed line around the replicas represents a subnet.

Figure 1 illustrates our approach at a high level. Starting from the left, an Arbitrum Nitro chain owner initializes the system in AnyTrust mode, providing the URL of the canister and the root key of the Internet Computer. During this initialization, Nitro contracts are deployed on Ethereum, with the SequencerInbox being the primary component of interest for data availability (DA) purposes. The interaction between Arbitrum, Ethereum, and the canister involves two key flows: store and get.

When a batch is ready for finalization and storage, the BatchPoster component signs and submits it to the canister for long-term storage, if signature is valid, the canister accepts it and generates a response. This response is threshold-signed by the subnet replicas and therefore guarantees that all honest nodes will store the data. Arbitrum uses this response to create its own certificate, called a DACert, which is the only data stored on Ethereum for a given batch. When a client needs to retrieve a batch from the canister, it first fetches the corresponding DACert from Ethereum. The client then queries the canister for the actual batch data. The canister responds with the data, a witness, and a certificate. The client can use the witness and certificate provided by the canister, along with the public key (IC root key) from the DACert, to validate and authenticate the retrieved data.

The integration into the Arbitrum workflow was seamless, thanks to the parity of signature schemes. DAC members in Arbitrum use BLS12-381 keys to authenticate themselves and the data they serve. Similarly, IC certification employs the same signature scheme, enabling us to replace the DAC with the IC with minimal intervention and changes to the upstream Arbitrum Nitro. In fact, it is potentially possible to integrate the solution without any modifications at all.

TABLE I
Comparison between IC, Celestia and Data Availability Committee

	IC	**Celestia**	**DAC**
Peer Join[1]	yes	yes	no
Estimated Costs[2]	$211	$3190	N/A
Block Time	< 0.5sec	< 7sec	N/A
Trust Assumption[4]	$< \frac{1}{3}n$	$< \frac{1}{3}n$	$< n-2$
Financial Security[5]	$2.4B	$1.7B	N/A

Lastly, we aim to demonstrate that our solution can stand against prominent alternatives. As a competitor, we have chosen Celestia, which, with its large market cap and substantial activity, is often considered the canonical choice for using a blockchain as a DA layer. To evaluate and compare, we have created an automated script [13] pulling data from a Celestia API aggregate Celenium [10] and using the official IC costs calculator library [12] to illustrate the costs differences. To ensure a reliable comparison, we focused on the storage requirements of Eclipse, the largest rollup utilizing Celestia for Data Availability. Eclipse hands over 2.27GB of data daily to Celestia, split in more than 10,000 times a day. We used these precise metrics to estimate the expected costs for performing the same operations on the Internet Computer using our solution, rather than Celestia. Our findings in Table I indicate that IC performs effectively as a DA layer and, in fact, surpasses Celestia in most metrics. It provides lower costs, enhanced financial security, while maintaining adequate distribution across more than one dozen nodes.

[1]Whether the storage layer has a protocol in place for node join/leave.
[2]USD cost for 30 days of storing 2.27GB
[3]Expected response time for a query
[4]Threshold of malicious nodes
[5]Amount of capital staked, correct for 2025-01-06

REFERENCES

[1] Vitalik Buterin, *Scaling Ethereum L1 and L2s in 2025 and beyond*, https://vitalik.eth.limo/general/2025/01/23/l1l2future.html, Accessed: April 21, 2025.

[2] *EIP-4844: Shard Blob Transactions, Proto-Danksharding*, https://www.eip4844.com/, Accessed: April 21, 2025.

[3] Mustafa Al-Bassam, *LazyLedger: A Distributed Data Availability Ledger With Client-Side Smart Contracts*, arXiv preprint arXiv:1905.09274, https://arxiv.org/pdf/1905.09274, Accessed: April 21, 2025.

[4] *Internet Computer*, https://internetcomputer.org, Accessed: January 6, 2025.

[5] *Nitro*, GitHub repository, https://github.com/OffchainLabs/nitro, Accessed: January 6, 2025.

[6] *Configure a Data Availability Committee*, https://docs.arbitrum.io/run-arbitrum-node/data-availability-committees/configure-dac, Accessed: January 10, 2025.

[7] *ICDA*, Proof of Concept GitHub repository, https://github.com/CommoDor64/nitro-icda-poc, Accessed: January 6, 2025.

[8] The DFINITY Team, *The Internet Computer for Geeks*, Cryptology *ePrint* Archive, Paper 2022/087, 2022. https://eprint.iacr.org/2022/087.

[9] *Deterministic Decentralization*, https://wiki.internetcomputer.org/wiki/Deterministic_Decentralization, Accessed: January 10, 2025.

[10] *Celenium.io*, https://celenium.io, Accessed: January 5, 2025.

[11] *Internet Computer Costs Calculator*, https://3d5wy-5aaaa-aaaag-qkhsq-cai.icp0.io/, Accessed: January 6, 2025.

[12] *Internet Computer Costs Calculator Library*, https://github.com/dfinity/icp-calculator, Accessed: January 6, 2025.

[13] *ICDA Celestia Compare*, https://github.com/CommoDor64/icda_solution_showdown/tree/icbc_2025_2, Accessed: January 6, 2025.